**Ágens megközelítés:**

* Sorolja fel a környezetet meghatározó tényezőket és ezek alapján definiálja a lehetséges környezeteket.

Teljesen/részben megfigyelhető:

* Ha az ágens szenzorai lehetővé teszik a teljes környezet megismerését minden egyes időpillanatban, akkor a környezetet teljesen **megismerhetőnek** mondjuk

Determinisztikus/Sztochasztikus:

* Ha a környezet a következő állapotát egyértelműen meghatározza az aktuális állapot és az ágens cselekedete, akkor a környezetet **determinisztikus.** A **sztochasztikusnál** nem határozza meg. **Stratégiai** a környezet, ha más ágensek cselekvéseit leszámítva determinisztikus.

Epizódszerű/Sorozatszerű:

* **Epizódszerű** környezetben az ágens tapasztalata atomi epizódokra bontható; minden egyes epizód az ágens észleléseiből és cselekedetéből áll. A következő epizód független az előzőtől.

Statikus/Dinamikus:

* Ha a környezet megváltozhat, amíg az ágens gondolkodik, akkor a környezet **dinamikus**. Ha a környezet időben nem változik, de az ágens teljesítménymértéke igen, akkor **szemidinamikus** a környezet.

Diszkrét/Folytonos:

* A diszkrét/folytonos megkülönböztetésnek alapja a környezet szerkezete, az idő múlása, az ágens észleléseinek és cselekedeteinek típusa.

Egy/Többágenses:

* Hány ágens található a környezetben?

Több ágens esetén lehet versengő vagy kooperatív.

A valós világ:

* részben megfigyelhető
* sztochasztikus
* sorozatszerű
* dinamikus
* folytonos
* többágenses

Ágensek típusai (egyszerűtől a bonyolultig):

* Egyszerű reflexiós
* Modellalapú reflexszerű
* Célorientált
* Hasznosságorientált
* Mi a racionális ágens?

Egy racionális ágens a legjobb várható kimenet érdekében cselkszik. Azt a műveletet választja, mely maximalizálja a várható teljesítményértéket adott érzékeléssorozathoz.

* Mitől függ egy ágens racionalitása (adott pillanatban)?

Az ágens sikerességétől, a környezet ismeretétől, a végrehajtható cselekvésektől, és az érzékelési sorozatától. TKBÉ – Teljesítmény, Környezet, Beavatkozók, Érzékelők.

* Mi a különbség és a hasonlóság az ágensprogram és ágensfüggvény között?

Az ágensfüggvény és program különbsége, hogy az egyik matematika leképezés míg a másik egy fizikai architektúrán fut, hasonlóság pedig az, hogy a program a függvényt valósítja meg

* Mi a specialitása az egyszerű reflex ágensnek?

A reflex ágens a környezet aktuális állapota alapján dönt, hogy mi legyen a következő művelet.

* Mi a specialitása a modellalapú reflexszerű ágensnek?

Az ágens képes az állapot tárolására, azaz van memóriája. A memóriában felépíti a modelljét.

* Mi a specialitása a célorientált ágensnek?

A célorientált ágens a környezet aktuális állapota és a kitűzött cél alapján dönti el, hogy mi lesz az a lépés, ami közelebb juttatja őt a célhoz.

* Mi a specialitása a hasznosságorientált ágensnek?

A környezet minden állapotához egy hasznossági értéket rendelünk, és a következő cselekvés kiválasztása esetén azt nézzük, hogy milyen teljesítményi szintet tudunk elérni, ez alapján választunk.

* Mi a specialitása a tanuló ágensnek?

A környezetből érkező inputot összeveti a teljesítményszabvánnyal és a tanuló elemnek továbbítja. Ezzel módosulhatnak a szabályok, célok vagy hasznosságfüggvények. Ha a tanuló úgy látja, hogy jó irányba módosultak ezek az értékek, akkor elmenti a módosításokat és a végrehajtható elem ezeket fogja alkalmazni.

**Keresések:**

* Mi a különbség az online és offline keresések között?

Az online keresésnél teljes tudás hiányában cselekszünk, menet közben új információkat szerzünk, ellen szemben az offline keresésnél egyszerre megismerjük az egész problémát, fejben megoldja, majd a kész megoldást átvezeti az adott környezetben.

* Milyen komponensei vannak egy jól definiált problémának keresési feladatok esetén?

Kezdeti állapot,

* rákövetkező függvény: megadja egy állapothoz, hogy milyen művelettel melyik állapotba lehet átjutni
* célteszt: célállapotok felsorolása, vagy egy feltétel megadása, amely pontosan a célállapotok esetén igaz
* útköltség: egy-egy cselekvéshez valamilyen költséget rendel, egy cselekvéssorozathoz ezen költségek összegét rendeljük, egy költség nem lehet negatív.
* A megoldás a cselekvéssorozat mely a kezdőállapotból a célállapotba vezet.
* Keresési feladat esetén hogyan adjuk meg a cselekvéseket?

Egy rákövetkező függvénnyel, amely megadja, hogy egy állapotból milyen művelettel melyik állapotba lehet átjutni, és az útköltséggel, mely minden cselekvéshez valamilyen költséget rendel.

* Mi a kapcsolat az állapottér és keresési fa között?

A keresési fa az állapottérből származtatható: a kezdeti állapot és az állapotátmenetfüggvény generálja

* Mit kell tárolni a keresési fa egy csúcsában?

Az állapotot, a csúcshoz tartozó szülőt azt, hogy milyen művelettel jutunk el hozzá, a mélységet és a gyökértől a csúcsig jutás költségét

* Mi a különbség egy állapot és egy keresési fa csúcsa között?

Egy állapot az a probléma egy állapota, a keresési fa csúcsa pedig csak tárolja ezt az állapotot.

* Adja meg az általános fakeresési algoritmust.

Megnézzük, hogy a fa üres-e, mert ha igen akkor nincs megoldás. Ha nem üres akkor kiválasztunk egy csúcsot a tárolóból, ami a fa egy levelét szimbolizálja. Ha ez a csúcs célállapot akkor visszaadjuk a gyökértől a csúcsig vezető utat, ha nem célállapot akkor addig határozzuk meg az összes rákövetkező állapotot és megnézzük mindegyikre, hogy célállapot-e.

* Adja meg az általános gráfkeresési algoritmust.

Megnézzük, hogy vannak-e állapotok, mert ha nem akkor nincs megoldás. Ha vannak akkor kiválasztunk egy csúcsot a tárolóból. A csúcsban szereplő állapotot felfedezettnek nyilvánítjuk és betesszük egy halmazban. Ha ez a csúcs célállapot akkor visszaadjuk a gyökértől a csúcsig vezető utat, ha nem célállapot akkor addig határozzuk meg az összes rákövetkező állapotot amíg meg nem találjuk a célállapotot, és olyan csúcsokra nem lépünk már amit bejártunk.

**Nem informált keresések:**

* Mit jelent az, hogy vak (nem informált) keresés?

Nincs információ az állapotokról a probléma definíciójában megadott információn kívül, csak annyiról, hogy melyik állapot célállapot és melyik nem. Működésük során két dolgot tehetnek: következő állaptok generálása, célállapot megkülönböztetése a nem célállapottól.

* Mi a perem? Adott adatszerkezetet peremként használva milyen keresési módszereket kapunk?

A perem egy gyűjtemény mely a legenerált, kifejtésre váró csomópontokat tartalmazza. A perem minden eleme egy levélcsomópont, azaz olyan csomópont, amelynek a fában nincsenek követői.

Szélességi keresésnél: FIFO sort használunk peremként, az új rákövetkezők a sor végére kerülnek, a sor első elemét terjesztjük ki.

Egyenletes költségű keresésnél: prioritás sort használunk, az útköltség szerint ideális esetben növekvő sorrendben.

Mélységi, mélységkorlátozott, iteratívan mélyülő mélységi keresésnél: LIFO vermet használunk, a rákövetkezőket a verem tetejére teszi, a legfelső elem kerül kiterjesztésre.

* Adja meg a lényeges különbségeket a különféle nem informált keresési módszerek között.

Szélességi keresés: a legkisebb mélységű még ki nem terjesztett csúcsot terjeszti ki.

Egyenletes költségű keresés: a legkisebb költségű még ki nem terjesztett csúcsot terjeszti ki, ha minden élköltség azonos, megegyezik a szélességi kereséssel.

Mélységi keresés: a legmélyebb ki nem fejtett csúcsot fejti ki.

Mélységkorlátozott: a mélységi keresés változata, a limit mélységben lévő csúcsoknak nincs ráközvetkezője.

Iteratívan mélyülő keresés: A mélységkorlátozott keresés limitjét iterációként eggyel növeli, és ezzel meghívja a mélységkorlátozott keresés függvényét.

* Melyek a teljes fakeresési módszere (milyen feltételek mellett)?

Szélességi keresés, ha a maximális elágazási faktor a kereső fában véges.

Egyenletes költségű keresés, ha az élköltségek pozitívok és nagyobbak epszilonnál.

Mélységkorlátozott keresés, ha a limit nagyobb vagy egyenlő, mint a legkisebb költségű megoldás mélysége.

Iteratívan mélyültő mélységi keresés, mindentől függetlenül teljes.

* Mely fakeresési módszerek optimálisak?

Szélességi keresés, ha minden lépés azonos költségű

Egyenletes költségű keresés

Iteratívan mélyülő mélységi keresés, ha az élköltség egy

* Adja meg a nem informált keresési módszereket a tárkomplexitásuk szerint növekvő sorrendben!

Iteratívan mélyülő mélységi keresés

Mélységkorlátozott

Mélységi keresés

Egyenletes költségű keresés

Szélességi keresés

* Adja meg a nem informált keresési módszereket az időkomplexitásuk szerint növekvő sorrendben!

Iteratívan mélyülő keresés

Egyenletes költségű keresés

Szélességi keresés

Mélységkorlátozott

Mélységi keresés

* A valós életben a szélességi keresés vagy az iterált mélységi keresés ad gyorsabban eredményt?

Az iteratívan mélyülő gyorsabban, mert d mélységben nem terjeszti ki a csúcsokat. A szélességi kereső módosítható, hogy a céltesztet akkor vizsgálja, amikor a csúcs generálódik.

* Melyek optimális gráfkeresési módszerek?

Az egyenletes költségű és a konstans lépésköltségű szélességi keresés, mert ez nem vét el optimális megoldást az újonnan felfedezett út eldobásával, ugyanis nem dobja el.

* Mikor szükséges gráfkeresést használni fakeresés helyett?

Ismétlődő állapotokat tartalmazó problémák esetén a gráfkeresés hatékonyabb. Ha az állapottér inkább gráfra hasonlít. A gráfkeresés az összes duplikált állapotot kiküszöböli.

**Informált keresések:**

* Melyek a legjobbat először keresés jellemzői?

A legkisebb értékű csomópontot fejtjük ki először, mert a kiértékelő függvény a céltól való távolságot méri, prioritásos sor segítségével implementálható, ami a peremet növekően rendezi. Kulcseleme a heurisztikus függvény, ahol h(n) = az n csomóponttól a célig vezető legolcsóbb út becsült útköltsége. Speciális esetei: mohó legjobbat először, A\* keresés.

* Milyen problémái vannak a mohó legjobbat először keresésnek?

Nem teljes, mert beragadhat, nem optimális, időbonyolultsága nem jó, de egy jó heurisztika drámaian felgyorsíthatja, tárbonyolultsága rossz, minden csúcsot a memóriában tart.

* Melyek az A\* keresés jellemzői?

Nem terjeszt ki olyan utakat, melyek drágák.

A célba vezető út becsült teljes költsége = útköltség a célig + a célig tartó út becsült költsége n-től.

Kiértékelt függvény: f(n) = g(n) + h(n)

h(n) <= mint a valós költség a célig

h(n) >= 0, így h (c) = 0 minden c cél esetén

A légvonalban mért távolság nem becsüli felül az úton mért távolságot

Az A\* fakeresés optimális

Teljes, ha nincs végtelen sok csúcs melyre f <= f(G), exponenciális az időbonyolultsága, minden csúcsot a memóriában tart.

* Mit jelent az elfogadható heurisztika?

h(n) egy elfogadható heurisztika, ha sosem becsüli felül a cél eléréséhez szükséges költséget.

* Mit garantál az elfogadható heurisztika fakeresés esetén?

Azt, hogy a fakereső A\* algoritmus optimális lesz

* Mit garantál az elfogadható heurisztika gráfkeresés esetén?

Azt, hogy a gráfkereső A\* algoritmus által kifejtett csomópontok sorozata f(n)-ben nem csökkenő.

f(n) = az n célba vezető út becsült teljes költsége.

* Mit jelent a konzisztens heurisztika?

Akkor konzisztens egy heurisztika, ha minden n csomópontra és annak egy tetszőleges cselekvéssel generált minden n’ utódcsomóponttól elért cél becsült költsége nem kisebb, mint az n’-be kerülés lépésköltsége és az n’ csomóponttól elért cél becsült költsége.

Tehát teljesül rá a háromszög egyenlőtlenség egy formája.

* Mit garantál a konzisztens heurisztika fa/gráf keresés estén?

Optimális megoldást A\* esetén.

* Miért fontos, ha egy heurisztika dominál egy másik heurisztikát?

Ha h2(n) >= h1(n) minden n csúcsra és mindkettő elfogadható, akkor h2 dominálja h1-et.

A domináns heurisztikát használva gyorsabb lehet megoldást találni.

* Miért lehetnek érdekesek a relaxált problémák keresésnél?

Egy probléma relaxált probléma, ha az operátorokra kevesebb megkötést teszünk, mint az eredeti problémába.

* A lokális algoritmusok miben hasonlítanak és miben térnek el a fa- és gráfkeresésektől?

A lokális algoritmusok lehetőség szerint a legjobb cél állapotot keresik.

Nem tárolják a célhoz vezető utat, csak egy aktuális állapotot vesznek figyelembe, és általában csak ennek az állapotának a szomszédjaira lépnek tovább. Nem szisztematikusak, kevés memóriát használnak, és sokszor nagy vagy végtelen térben is elfogadható megoldást adnak.

* Adja meg a hegymászó algoritmust!

Bemenete: egy probléma

Kimenete: egy lokális maximumot jelentő állapot

A kezdőállapot lesz az aktuális változó, és elkezdünk egy végtelen ciklust. Végig megyünk az aktuális állapot összes szomszédján, és kiválasztjuk azt, amelyhez a célfüggvény maximális, ezt kell összevetni az aktuális állapottal, és ha nem jobb, mint az aktuális, akkor visszaadjuk az aktuális állapotot, mint lokális maximum, ha szomszédhoz jobb a célfüggvény értéke, mint az aktuális állapothoz, akkor az aktuális állapot a szomszéd lesz.

* Milyen problémái vannak a hegymászó algoritmusnak?

Csak lokális maximumot tud generálni, globálisat nem.

Lokális maximumokon, hegygerinceken, fennsíkon az algoritmus beragadhat.

* Mi a lényege a genetikus algoritmusnak?

Ez a sztochasztikus nyaláb keresésnek egy olyan variánsa, ahol a követő állapotokat nem egy állapot módosításával, hanem két szülő állapot összekombinálásával állítjuk elő.

Állaptok halmaza lesz a kezdeti populáció

A célfüggvény megadja az egyes állaptok kívánatosságat.

Erre alapozva választjuk ki az állapot-párokat, ezután a két állapotot keresztezzük, majd ennek eredményét egy kicsit megváltoztatjuk, ez lesz a mutáció. Az így kapott állapotok alkotják az új populációt

* Mi a lényege a szimultán hűtésnek?

A lényege, hogy kiszabaduljanak a lokális maximumból, azzal, hogy rossz lépéseket megengedünk, de ezek méretét és gyakoriságát sorra csökkentjük.

A kezdeti aktuális állapot a kezdőállapot, elindítunk egy végtelen ciklust.

Megnézzük az aktuális állapot egy random szomszédját, és ha jobb az értéke akkor átlépünk oda, ha rosszabb akkor a véletlen dönt arról, hogy átlépünk e. A nagy hőmérséklet lehetővé teszi a rosszabb irányba történő lépést, ahogy a hőmérséklet csökken, ezt egyre kevésbé engedi. A hőmérsékletet egyre csökkentjük, és ha egy adott értékre lecsökken, akkor kilépünk a végtelen ciklusból, és az aktuális állapotot adjuk vissza megoldásként.

* Mi a lényege a minimális konfliktusok algoritmusának?

Amikor egy változónak új értéket választunk, azt az értéket választjuk ki, amelyik a legkevesebb konfliktust eredményezi más változókkal, ezt nevezzük minimális konfliktusok heurisztikájának.

**Kényszer-kielégítés:**

* Mi a különbség egy általános fakeresési, valamint egy kényszer-kielégítési probléma között?

Az állapot bármilyen adatstruktúra ábrázolhatja, és csak az állapotátmenet, heurisztika és célállapot van implementálva.

A kényszer-kielégítési problémát több változóval írhajuk le, melyek egy tartományból jönnek, minden változónak lesz egy értéke, ez lesz egy kényszer. Az állapotokat a Di tartományból származó Xi változókkal definiáljuk.

* Mivel lehet megadni egy kényszer-kielégítés feladatot?

Változók (X1, X2, … Xn) és kényszerek (C1, C2, … Cn) halmazaival. Minden Xi változó esetén adott a lehetséges értékek egy nem üres Di tartománya. Egy problémaállapotot az definiál, hogy vagy néhány, vagy az összes változóhoz értéket rendelünk. Egy hozzárendelés konzisztens, ha egy kényszert sem sért meg. Teljes az a hozzárendelés, amelyben minden változó szerepel, és a problémának megoldása, ha miden kényszert kielégít. Néhány kényszerkielégítési probléma azt is igényli, hogy a megoldás egy célfüggvényt maximalizáljon.

* Mit jelent a kemény és puha kényszer? (Abszolút- és preferenciakényszer)

A kemény kényszernek mindenképpen eleget kell tenni, a puha kényszer elhagyható, hogyha másképpen nem megoldható a probléma.

* Hogyan lehet megoldani egy kényszer-kielégítési problémát?

Kényszergráfra használatos módszerek: standard (inkrementális keresési módszer), visszalépéses keresés, minimális konfliktusok.

* Milyen kényszerek lehetnek egy kényszer-kielégítési problémában?

Kemény kényszerek:

unáris: egy változót tartalmaz

bináris: két változót tartalmaz

magasabb rendű kényszer: legalább három változót tartalmaz

preferencia – puha kényszer

* Mit tartalmaz a kényszergráf, és mit jelölnek egyes alkotórészei?

A kényszergráf a problémát reprezentálja, csúcsai a változók, amire alkalmazzuk a kényszereket, az élek pedig a kényszerek.

* Hogyan működik a visszalépéses keresés egy kényszer-kielégítési probléma esetén?

Egy olyan mélységi keresés, ahol egyszerre egy változó kap értéket. Egy adott csúcsban csak egy változó értékadásaival folytathatjuk.

* Milyen heurisztikákkal lehet gyorsítani a visszalépéses keresésen? Ismertesse az egyes módszerek lényegét!

Legkevesebb fennmaradó érték: azt a változót választjuk, mely a legkevesebb megengedett értékkel rendelkezik. (legkorlátozottabb változó)

Fokszám heurisztika: holtverseny esetén dönt a legkevesebb fennmaradó érték változóiról: válasszuk azt a változót, amely legtöbbször szerepel a hozzárendeletlen változókra vonatkozó kényszerekben.

Legkevésbé korlátozó érték: értékek sorrendje: azt az éréket választjuk, mely a legkevesebb éréket zárja ki a hozzárendeletlen változóknál

Előre néző ellenőrzés: tartsuk nyilván a hozzárendeletlen változók lehetséges értékeit, és ha valamely változónak már nincs lehetséges értéke, állítsuk le a keresést.

Élkonzisztencia: a kényszerek terjeszkedésének legegyszerűbb formája, ha minden élt konzisztenssé teszünk: X 🡪 Y él konzisztens, ha X minden x értékéhez van y eleme Y megengedett érték.

* Mi a különbség az előre néző ellenőrzés (forward checking) és a kényszerek terjesztése (constaint propagation) között?

A kényszerek terjesztése lokálisan érvényesíti a kényszereket, hamarabb észreveszi a hibákat, az előre néző ellenőrzés csak a közvetlen következményeket tudja felderíteni.

* Hogyan használható az él-konzisztencia? Milyen korlátai vannak?

Minden élt konzisztensé teszünk, X 🡪 Y él konzisztens, ha X minden x értékéhez van y elem Y megengedett érték. Ha egy változó értékét töröljük, a szomszédait is újra kell vizsgálni.

**Kétszemélyes játékok:**

* Hogyan kapcsolhatóak a kétszemélyes játékok a keresési problémákhoz?

Meg kell határozni a legjobb lépést az ellenfél minden lehetséges lépésre, megadott idő alatt, így csak egy közelítő megoldásra van szükség.

* Hogyan működik a minmax keresés?

Az algoritmus a lehető legjobb lépéshez tartozó operátort adja vissza, vagyis ahhoz a lépéshez tartozó operátort, amelyik a legnagyobb hasznossági értékkel rendelkező eredményre vezet, feltételezve, hogy az ellenfél úgy játszik, hogy minimalizálja a hasznossági értéket. A max-érték és min-érték függvények végig mennek a teljes játékfán egészen a levelekig, hogy meghatározzák a csomópont felfelé terjesztett értékét.

* Használható-e minmax keresés akkor, amikor a játékfa nem fér el egyszerre a memóriában? Milyen módon?

Igen, mert a minmax keresés mélységi keresést használ, ezért csak elég azokat a csúcsokat a memóriában tartani, amelyeket éppen használunk.

* Mit jelöl az alfa és béta az alfa-béta nyesés esetén?

Az alfa a legjobb érték max szerint, amit eddig az aktuális úton találtunk, a béta a legjobb érték min szerint, amit eddig az aktuális útvonalon találtunk.

* Hogyan kezelhetjük a véletlent is tartalmazó játékokat?

Egy expectiminmax algoritmussal, amely ugyanaz, mint a minmax függvény csak a véletlent is kezeli, és lesz egy harmadik típusú „chance” csúcsa, ilyenkor az elérhető értékek valószínűséggel súlyozott értékét kell venni.

* Hogyan kezelhetjük a nem teljesen informált játékokat?

A lehetséges eseteket egy véletlen mintából jól megközelítjük, teszteljük, hogy a minta esetén melyik lépés a legjobb és azt választjuk.

**Bizonytalanság kezelése:**

* Mit jelent az, hogy priori?

Az előzetes valószínűség elnevezése, jellemzően tapasztalatból nyert elképzelés. Mielőtt a tények birtokába jutunk előzetes, priori, vagy feltétel nélküli valószínűségről beszélünk.

* Mit jelent a feltételes valószínűség?

Tudjuk valaminek a valószínűségét feltéve, ha egy rá hatással lévő esemény már bekövetkezett.

P (a | b) = P(ab) / P(b)

* Adja meg a sorozatszabályt!

P (ab) = P (a | b) \* P(b) = P (b | a) \*P(a)

* Mit jelent, hogy két valószínűségi változó független?

Két változó független, ha bekövetkezésük valószínűségét nem változtatja meg a másik változó hamarabbi bekövetkezése. Tehát A és B független ha

P (A | B) = P(A) \* P(B)

* Mit jelent, hogy két valószínűségi változó feltételesen független?

Egy adott feltétel menetén szabadulhatunk meg egy másik feltételtől, tehát van függetlenség, de csak egy feltétel teljesülésekor.

P (X, Y | Z) = P(X|Z) \* P(Y|Z), tehát Z esetén X és Y független

* Hogyan adható meg egy Bayes-hálózat?

A Bayes-hálózat valójában egy irányított gráf. A gráf csúcsait a valószínűségi változók az élei azt mutatják meg, hogy mely valószínűségi változók között van kapcsolat, és ennek milyen az iránya. Egy-egy élhez valószínűségi információk is tartozhatnak, a csúcsba vezető élek együtt határozzák meg ezt. Így a valószínűségi változókhoz feltételes valószínűségi táblázatok tartoznak.

* Adja meg az együttes valószínűség meghatározására vonatkozó láncszabályt!



**Valószínűségi következtetés:**

* Hogyan alakul ki a feltételes valószínűség táblázat Bayes hálók esetén?

Minden Xi csomóponthoz tartozik egy P (Xi | Szülők (Xi)) feltételes valószínűségi eloszlás, mely megadja a szülők hatását a csomóponti változóra.

* legegyszerűbb esetben ezt egy feltételes valószínűségi táblázat segítségével adjuk meg
* Mit értünk valószínűségi következtetés alatt?

Kiszámítja a célváltozók egy halmazának a posteriori valószínűség-eloszlását egy adott megfigyelt esemény esetén.

Azaz bizonyítékváltozók egy halmazához történő érték hozzárendelés esetén.

**Megfigyelésen alapúló tanulás:**

* Mi az a hipotézis függvény és mit jelent, amikor a modellünk konzisztens?

Adott az f-re vonatkozó minták egy halmaza, ennek alapján határozzunk meg egy h függvényt,

amely közelíti f-et. A h függvényt hipotézisnek (hypothesis) nevezzük.

h akkor konzisztens, ha minden minta esetén megegyezik f-el

* Hogyan definiálhatjuk a döntési fát? (bemenete, kimenete, paraméterei)

A hipotézis egy lehetséges reprezentációja

Bemenete: attribútumokkal leírt objektum, lehetnek diszkrétek vagy folytonosak

Kimenete: egy döntés

* diszkrét: osztályozás
* folytonos: regressziós

Tesztsorozat végrehajtása

* Hogyan konstruálhatjuk meg a döntési fánkat?

Válasszuk ki a legjobb attribútumot a pozitív és negatív példák szétosztására

Ha minden példa pozitív/negatív – a válasz Igen/Nem

Nem maradt példa valamelyik válaszhoz – a válasz a szülőcsomópontban a többségi válasz

Ha nincs tesztelendő attribútum:

* a pozitív és negatív példák nem választhatóak szét
* zajos adat vagy nemdeterminisztikus problématér
* többségi szavazás
* Írja le, hogy mit jelent az információ nyereség és hogyan használhatjuk a döntési fa felépítéséhez.

Információnyereség: információtartalom, maradék információ szükséglet

A döntési fa tanulásánál használhatjuk

* Írja le, hogyan és milyen módszerekkel lehet kiértékelni egy betanított modell pontosságát.

Gyűjtsünk egy nagy példahalmazt

Osszuk két diszjunkt halmazra

* tanító halmaz (training set)
* teszthalmaz (test set)

Alkalmazzuk a tanító algoritmust a tanító halmazokon, és generáljunk egy h hipotézist Mérjük meg a teszthalmazon, hogy a h hipotézis a halmaz hány százalékára ad helyes osztályba sorolást Ismételjük meg a lépéseket különbözően választott tanító halmazokra

**Statisztikai tanulás:**

* Bayes tanulás esetén mit nevezünk tényeknek és mi lesz a hipotézisünk?

Tények: adott területet leíró valószínűségi változók konkrét megvalósulása

* d = d1, … dn

Hipotézis: elmélet: hogyan működik a világ?

H – hipotézis változó h1, h2, … értékek P(H) a priori eloszlása

* Mi a maximuma a posteriori hipotézisnek és hogyan számolhatjuk? MAP

A hipotézisek tere gyakran kezelhetetlenül nagy egyetlen, a legvalószínűbb hipotézis alapján végezzük a predikációt.

* azon hi alapján, mely maximalizálja P (hi | d) -t
* maximalizálja P (d | hi ) \* P (hi )
* utóbbi tekinthető azon bitek számának, ami kódolja a mintát az adott hipotézis esetén + kódolja a hipotézist
* Mi az a maximum likelihood hipotézis és hogyan számolhatjuk?

Feltétel: egyenletes P(H) priori eloszlás

Statisztikában nagyon elterjedt, standard módszer

Hasznos, ha a hipotézisek komplexek, kezdeti eloszlást nehéz meghatározni

Nagy adathalmaz esetén megközelíti a Bayes- és MAP tanulást

* Mutassa be a naiv-Bayes modell működésének alapját!

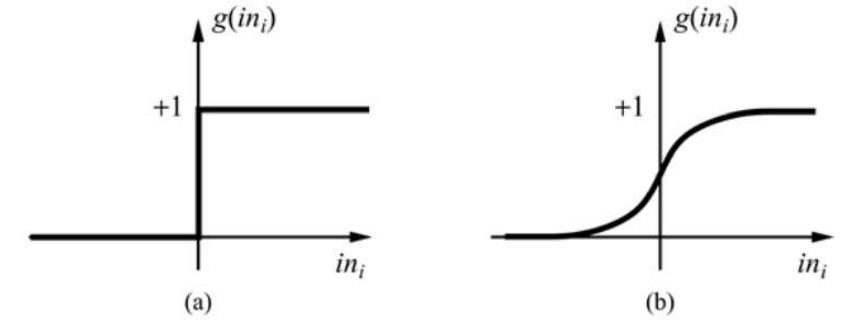
A modellben a C „osztályváltozó” a gyökér. Míg Xi attribútumváltozók a levelek. A modell azért „naiv”, mert feltételezi, hogy adott osztály mellett az attribútumok feltételesen függetlenek egymástól.

**Neurális hálók:**

* Mutassa be hogyan működik a neurális háló egy neuronja!

A neurális háló egy egysége mely kap egy bementet majd, ha a bemenet érékek súlyozott összege meghaladj egy küszöböt akkor „tüzel”, mint egy biológiai neuron.

* Milyen aktivációs függvényeket ismer és mik a különbségek?
* A küszöb aktivációs függvény, amely 1-et ad a kimeneten, ha a bemenete pozitív, különben pedig 0-t.
* A szigmoid, vagy logisztikus függvény.



* Milyen neurális háló struktúrákat ismer és mik a különbségek?

A neurális hálóstruktúrák két fő csoportja: a hurokmentes vagy előre csatolt hálók és a visszacsatolt, vagy rekurrens hálók.

Az előre csatolt háló a pillanatnyi bemenet függvényét reprezentálja, azaz nincs semmilyen más belső állapota, csak maguk a súlyok. A rekurrens háló viszont a kimeneteit visszacsatolja a bemeneteire.

Ezenfelül a háló egy adott bemenetre adott válasza a kezdeti állapotától függ, amely a korábbi bemenetektől függhet. Ennélfogva a rekurrens hálók rövid távú memóriát is biztosíthatnak. Ezáltal érdekesebbé válnak, mint agymodellek, de egyben nehezebben is érthetőek.

* Egy egyszerű példán keresztül mutassa be, hogyan működik egy egyrétegű előre csatolt neurális háló!

Az összes bemenet közvetlenül a kimenetre kapcsolódik.

Minden kimeneti egység független a többitől, így elég egyenként vizsgálni

A súlyok módosításával az output-függvény helye, alakja, iránya változtatható

* Hogyan tudjuk tanítani a neurális hálónk súlyait? Milyen módszer ismert rá és mi a módszer alapelve?

A tanulás a tanulóhalmazon mért hiba csökkentése a súlyok módosításával x input, y output esetén a négyzetes hiba

Gradiensalapú optimalizálás, minden súlyra vonatkozó parciális derivált kell

Súlyfrissítés: Ha a hiba pozitív, pozitív bemenet súlyait növelni, negatív bemenet súlyait csökkenteni kell.

* Milyen típusú tanulási módszereket ismer és mik a különbséget? Miként különböznek diszkrét és folytonos esetben?

Többrétegű előre csatolt neurális hálók (TEN): a rétegek rendszerint teljesen összekötöttek

A rejtett rétegek és egységek számának meghatározására nincs módszer

Hiba-visszaterjesztéses tanulás (back-propagation)

Tanulási görbe: a tanuló algoritmus minden egyes mintára kiszámolja a hálót, és kissé módosítja a súlyokat

* Mutassa be a konvolúció műveletét!

Azt mondjuk, hogy a h függvény az f és a g függvények konvolúciója, ha

egy dimenzióban, illetve

két dimenzióban

**Megerősítéses tanulás:**

* Hogyan szól a megerősítéses tanulás definíciója?

A megerősítéses tanulás a gépi tanulás egy olyan területe, mely azzal foglalkozik, hogy intelligens ágens miként cselekedjen egy környezetben, hogy maximalizálja az összjutalmat.

* Írja le koncepcionális szinten a megerősítéses tanulás működését. Miként interaktál az ágens a környezetével?

Az ágens egy adott környezetben, adott S állapoton megtesz egy lépést, ami alapján kap egy jutalmat és egy új, a lépés által keletkezett állapotot. Ezt a jutalmat maximalizáljuk. Tehát az interakció a környezettel a lépés és a környezet visszafele ad jutalmat.

S = State, a = Action, r = reward, S’ next state

* Mit értünk Markov döntési folyamat alatt és mik az alapvető alkotó elemei?

A jelenlegi állapot csak az előző állapottól függ.

Alkotó elemek:

* S – States = Megfigyelt világ állapota, lehet részlegesen megfigyelhető, véges / végtelen, diszkrét / folytonos
* A – Actions = Megadja az ágens mit léphet adott állapotban, lehet véges / végtelen, diszkrét / folytonos
* R – Reward function
* P – State transition probabilites
* Y – Discount factor = azonnal, vagy hosszútávon maximalizáljuk a jutalmat
* Mit értünk policy függvény alatt a megerősítéses tanuláshoz kapcsolódóan?

Egy függvény, mely minden állapothoz egy lépést / cselekvést rendel. Lehet determinisztikus / sztochasztikus.

Azt írja le milyen stratégiát hajt végre az ágens.

* Mit értünk value és quality függvény alatt a megerősítéses tanuláshoz kapcsolódóan?

Value: Egy érték mely alapján össze tudunk hasonlítani állapotokat, azt jelöli az adott állapotban mennyire jó az ágens helyzete.

Quality: Egy érték mely alapján össze tudunk hasonlítani lépéseket / cselekvéseket, hogy melyiket válassza az ágens

* Hogyan tudjuk összegezni a lépések után járó jutalmakat a megerősítéses tanulás visszacsatolásához?

𝑅(𝜏) = 𝑟𝑡+1 + γ ∗ 𝑟𝑡+2 + γ2 ∗ 𝑟𝑡+2 + γ3 ∗ 𝑟𝑡+3 + …

γ = discount faktor

γ = 0 🡪 azonnali jutalmak érdekelnek

γ = 1 🡪 azonnali és jövőbeli jutalmak ugyanannyira számítanak

* Hogyan működik a gamma (γ) jutalmak összegzésekor?

γ = discount faktor

γ = 0 🡪 azonnali jutalmak érdekelnek

γ = 1 🡪 azonnali és jövőbeli jutalmak ugyanannyira számítanak

* Mit jelent a felderítés és kiaknázás a megerősítése tanuláshoz kapcsolódóan?
* Felderítés

Ágens új infót, állapotokat, cselekvéseket keres a környezetében. Aktívan kísérletezik, jutalmakat, következő állapotokat vizsgál.

* Kiaknázás

Ágens az eddig megszerzett információkra támaszkodva választ.

Már ismert, jól teljesítő állapotokat választ.

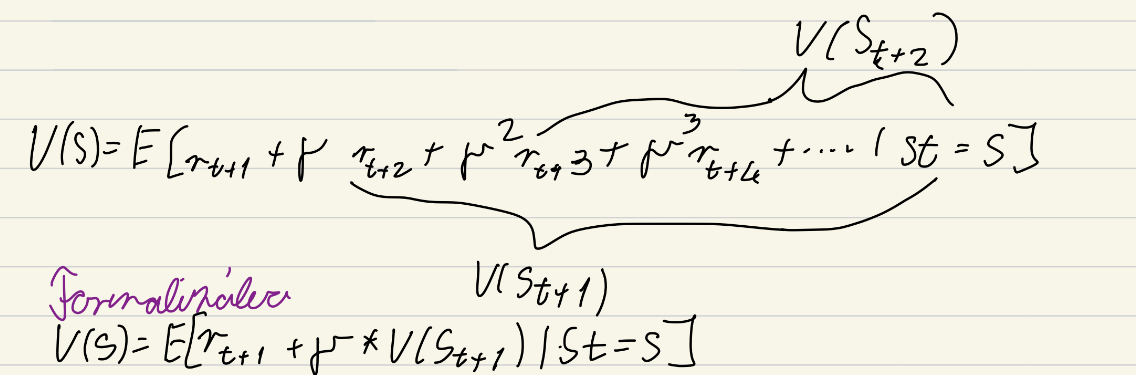
* Hogyan szabályozza az ε értéke a felderítés és kiaknázás közötti eloszlását?

Epszilon – Mohó stratégia, ahol a kiaknázás esélye

1 – ε ε [0, 1]

ε értéke folyamatosan csökken

* Írja le a Bellman egyenlet lényegét és formalizálja is



* Ez azt jelenti, hogy egy állapot értéke az azonnali jutalom 𝑟𝑡+1 plusz a következő állapot diszkontált értéke, ami γ ∗ 𝑉(𝑠𝑡+1)
* Hogyan használható a TD-tanulás a value és quality függvények tanításához?

A képen szöveg, kézírás, Betűtípus, kalligráfia látható

Automatikusan generált leírás